GitHub Actions: Using Pre-Built Actions

From Writing Commands to Using Ready-Made Tools

What You Already Know

Your Hello World Action:

```
- name: Greet the world
  run: echo "Hello from GitHub Actions!"    ← You wrote this command

- name: Show date
  run: date                                ← You wrote this command
```

Key Points:

- ✅ `run:` = Write your own commands
- ✅ **Simple commands** like `echo`, `date`, `ls`
- ✅ **Works great** for basic tasks

But what about complex tasks?

The Problem with Complex Tasks

What if you want to:

- 📥 **Download your code** from GitHub
- 🏗️ **Install Hugo** with all dependencies
- 🌐 **Deploy to GitHub Pages** with proper permissions
- 🔧 **Configure SSL certificates**
- 📊 **Set up caching** for faster builds

Writing this yourself:

```
# This would be 50+ lines of complex commands
curl -L https://github.com/gohugoio/hugo/releases/...
tar -xzf hugo_extended_...
chmod +x hugo
./hugo version
# ... and 46 more lines of configuration
```

😱 **Too complicated!**

Solution: Pre-Built Actions

Think of GitHub Actions Marketplace like an App Store:

📱 **App Store for Your Phone:**

- Don't write your own camera app
- Download Instagram (someone else built it)
- Just use it!

🤖 **Actions Store for GitHub:**

- Don't write your own deployment code
- Use `actions/deploy-pages@v2` (someone else built it)
- Just use it!

Key Difference:

```
run: echo "Hello"           ← You write the command
uses: actions/checkout@v4   ← You use someone else's tool
```

Understanding `uses` vs `run`

`run:` = Write Your Own

```
- name: Say hello
  run: echo "Hello World!"                    ← A simple command you write

- name: List files
  run: ls -la                                ← Another command you write

- name: Multiple commands
  run: |                                     ← Multiple commands you write
    echo "Starting..."
    date
    echo "Done!"
```

`uses:` = Use Pre-Built Tool

```
- name: Download code
  uses: actions/checkout@v4                ← Pre-built tool

- name: Deploy website
  uses: actions/deploy-pages@v2            ← Pre-built tool
```

Understanding Action Names

Action Name Format:

```
uses: actions/checkout@v4
        ↑        ↑          ↑
    owner/   name    version
```

Breaking it Down:

- `actions` = The organization/company that made it
- `checkout` = What the action does
- `@v4` = Version number (like app version 4.0)

Examples:

```
uses: actions/checkout@v4         # GitHub's official code downloader
uses: actions/setup-node@v3       # GitHub's Node.js installer
uses: actions/deploy-pages@v2     # GitHub's website deployer
```

The Three Key Actions for Hugo

1. actions/checkout@v4

```yaml
- name: Checkout
  uses: actions/checkout@v4
```

What it does:

```
📥 Downloads your repository code
📁 Makes it available in the virtual machine on GitHub
✅ Like "git clone" but automatic
```

Why you need it:

- GitHub Actions starts with an **empty computer**
- You need to **get your Hugo files** first
- This action **downloads everything** from your repository

2. actions/configure-pages@v3

```yaml
- name: Setup Pages
  id: pages
  uses: actions/configure-pages@v3
```

What it does:

```
🔧 Prepares GitHub Pages hosting
🌐 Sets up the website URL
🔒 Configures permissions
✅ Gets everything ready for deployment
```

Why you need it:

- GitHub Pages needs **special setup**
- This action **handles all the complex configuration**
- Without it, deployment will **fail**

3. actions/deploy-pages@v2

```yaml
- name: Deploy to GitHub Pages
  id: deployment
  uses: actions/deploy-pages@v2
```

What it does:

```
🚀 Takes your built website
📤 Uploads it to GitHub Pages
🌐 Makes it live on the internet
✅ Your website goes online!
```

Why you need it:

- This is the **magic step** that publishes your site
- Handles all the **complex deployment process**
- Without it, your site stays **only on the build machine**

Understanding `with:` Parameters

Some Actions Need Extra Information:

```
- name: Checkout
  uses: actions/checkout@v4
  with:                                          ← Extra settings
    submodules: recursive                        ← Download submodules too
```

Think of `with:` like Settings:

📱 Installing an App:

```
Install Camera App
Settings:
  - Quality: High
  - Flash: Auto
  - Location: Enabled
```

🤖 Using an Action:

```
uses: actions/checkout@v4
with:
  submodules: recursive
  fetch-depth: 0
```

Understanding `id:` References

Why Some Actions Have `id:`:

```
- name: Setup Pages
  id: pages                                 ← Give this step a name
  uses: actions/configure-pages@v3

- name: Build with Hugo
  run: |
    hugo --baseURL "${{ steps.pages.outputs.base_url }}/"
                    ↑
                    Use output from the "pages" step.
```

Think of `id:` like Variables:

```
Step 1: pages = Setup Pages (saves website URL)
Step 2: Use the URL from the pages step
```

Common pattern in Hugo deployment!

Complete Example: Hello World vs Hugo

Your Simple Hello World:

```
name: My First Action
on: push
jobs:
  say-hello:
    runs-on: ubuntu-latest
    steps:
      - name: Greet the world
        run: echo "Hello!"           ← Simple command
```

Professional Hugo Deployment:

```
name: Deploy Hugo Site
on: push
jobs:
  deploy:
    runs-on: ubuntu-latest
```

Why Use Pre-Built Actions?

Benefits:

✅ **Tested by thousands** of developers

✅ **Maintained by experts** who understand the complexity

✅ **Updated automatically** when GitHub changes

✅ **Save time** - no need to write complex code

✅ **More reliable** than custom scripts

Comparison:

```
Writing your own deployment: 200+ lines, many bugs
Using actions/deploy-pages@v2: 1 line, works perfectly
```

Professional Standard:

- Real companies use pre-built actions
- Writing everything yourself is **not professional**
- Using the right tools shows **good judgment**

Where to Find Actions

GitHub Actions Marketplace:

🌐 **Website:** https://github.com/marketplace?type=actions

Popular Categories:

- **Deployment:** Deploy to various platforms
- **Testing:** Run automated tests
- **Security:** Scan for vulnerabilities
- **Notifications:** Send alerts
- **Utilities:** Common development tasks

How to Choose:

✅ **Official actions** (made by `actions/`)
✅ **High star rating** (popular = tested)
✅ **Recent updates** (actively maintained)
✅ **Good documentation** (easy to use)

Common Hugo Actions Workflow

Complete Hugo Deployment Pattern:

```yaml
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      # 1. Get the code
      - name: Checkout
        uses: actions/checkout@v4
        with:
          submodules: recursive

      # 2. Install Hugo
      - name: Setup Hugo
        uses: peaceiris/actions-hugo@v2
        with:
          hugo-version: 'latest'
      # 3. Setup Pages hosting
      - name: Setup Pages
        uses: actions/configure-pages@v3

      # 4. Build the site
      - name: Build
        run: hugo --minify

      # 5. Deploy online
      - name: Deploy
        uses: actions/deploy-pages@v2
```